NPS ARCHIVE
2000.06
LOPEZ, B.

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY CA 93943-5101

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

AN INVESTIGATION AND ASSESSMENT OF LINUX IPCHAINS AND ITS VULNERABILITIES WITH RESPECT TO NETWORK SECURITY

by

Bryan S. Lopez

June 2000

Thesis Advisor: Raymond F. Bernstein Jr.

Approved for public release; distribution is unlimited.

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY CA 93943-5101

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2000	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE
An Investigation and Assessment of Linux Ipchains and its Vulnerabilities with Respect To Network Security

5. FUNDING NUMBERS

6. AUTHOR(S)
Bryan S. Lopez

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this research paper are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

This research thesis formulates a survey of network security and IPChains, the Linux firewall. It provides a detailed description of prominent network security procedures in use today. This paper falls directly in line with the goals of Executive Order 13010, the President's Critical Infrastructure Protection Plan, supports the goals of the National Security Agency's SIGINT Business Plan and the goals of both the Unified and Maritime Cryptologic Architecture. It will aid in the development of the problem solving efforts of the national cryptologic organization and be used to provide critical intelligence support to the Operational command and the national intelligence community.

14. SUBJECT TERMS
Linux, Network Security, Ipchains, Unified Cryptologic Architecture

15. NUMBER OF PAGES
137

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18, 298-102

Approved for public release; distribution is unlimited.

AN INVESTIGATION AND ASSESSMENT OF LINUX IPCHAINS AND ITS VULNERABILITIES WITH RESPECT TO NETWORK SECURITY

Bryan S. Lopez
Lieutenant, United States Navy
B.A., University of New Mexico, 1990
M.S., Troy State University, 1992

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2000

NPS ARCHIVE
2000.06
LOPEZ, B.

~~Thesis
L82244
c.1~~

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCH
MONTEREY CA 93943-5101

ABSTRACT

This research thesis formulates a survey of network security and IPChains, the Linux firewall. It provides a detailed description of prominent network security procedures in use today. This paper falls directly in line with the goals of Executive Order 13010, the Critical Infrastructure Protection Plan, supports the goals of the National Security Agency's SIGINT Business Plan and the goals of both the Unified and Maritime Cryptologic Architecture. It will aid in the development of problem solving efforts of the cryptologic organization and be used to provide critical intelligence support to the Operational command and the national intelligence community.

TABLE OF CONTENTS

I. LINUX 1

A. GOALS OF RESEARCH 1
B. WHY LINUX? 2
C. INTRODUCTION 5

II. FIREWALLS 9

A. OVERVIEW 10
B. DMZ 12

III. TYPES OF FIREWALLS 15

A. OSI MODEL 15
B. SCREENING ROUTER (PACKET FILTERS) 16
C. PROXY SERVER GATEWAYS 17
D. CIRCUIT-LEVEL GATEWAY 17
E. APPLICATION-LEVEL GATEWAY 18

IV. SECURITY CONSIDERATIONS 21

A. STATEFUL INSPECTION TECHNIQUES 21
B. SECURITY POLICIES 22
C. SERVICE PORTS: KEY ACCESS POINTS 24

V. PACKETS: IP NETWORK MESSAGES 27

A. INTERNET CONTROL MESSAGE PROTOCOL (ICMP) 28
B. USER DATAGRAM PROTOCOL (UDP) 33
C. TRANSMISSION CONTROL PROTOCOL (TCP) 34

VI. TYPES OF ATTACK 37

A. ICMP REDIRECTS AND BOMBS 37
B. DENIAL OF SERVICE 38
C. SMTP SESSION HIJACKING 39
D. EXPLOITING BUGS IN APPLICATION 39
E. BUGS IN OPERATING SYSTEMS 39
F. SYN ATTACK 40

VII. RESEARCH 43

A. APPROACH 43
B. IPCHAINS 44
C. A TYPICAL ATTACK 46
D. HACKING TOOLS 47
1. NMAP 47
2. RED BUTTON 53
3. SHADOW ADVANTIS ADMINISTRATOR 58
4. SUB 7 60

VIII. SYSTEM CONFIGURATION 63

A. HARDWARE 63
B. SOFTWARE 65
C. ROUTING 65
D. SECURITY ASSESSMENT WITHOUT FIREWALL 67
E. BUILDING THE FIREWALL 72
F. SECURITY ASSESSMENT WITH FIREWALL 82

IX. CONCLUSIONS 89

A. SUMMARY OF FINDINGS 89

APPENDIX A: IPCHAINS RULES AND ACTIONS 91

1. IPCHAINS ACTIONS 91
2. IPCHAINS PARAMETER TYPES 92
3. OPTION CHOICES 92
4. IPCHAINS RULE SPECIFICATIONS 93

APPENDIX B: NMAP 97

APPENDIX C: GLOSSARY 113

LIST OF REFERENCES 123

INITIAL DISTRIBUTION LIST 127

I. LINUX

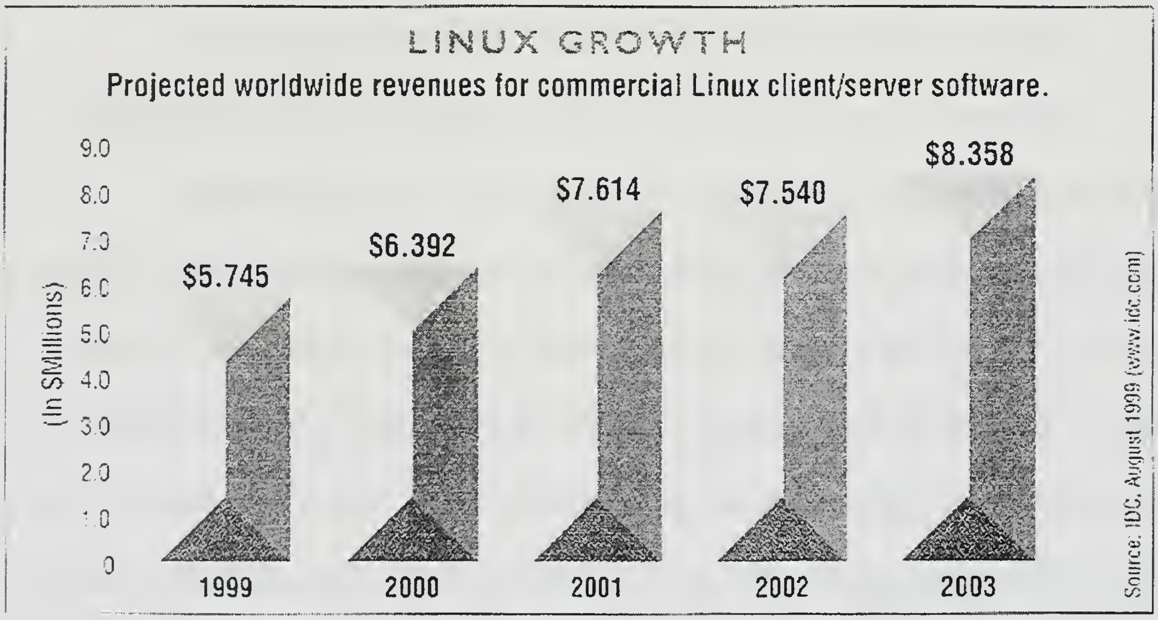
A. GOALS OF RESEARCH

This research thesis was conducted as a means of investigating the network security characteristics associated with the Linux operating system. A close examination of IPChains, the Linux firewall, was undertaken in pursuit of determining its relative vulnerability vis-a-vis the other popular networking software, Microsoft's Windows NT. This research falls directly in line with the goals of Executive Order 13010, the Critical Infrastructure Protection Plan and supports the goals of the National Security Agency's (NSA's) SIGINT Business Plan and the goals of both the Unified and Maritime Cryptologic Architecture. It will aid in the development of the problem solving efforts of the national cryptologic organization and be used to provide critical intelligence support to the Operational command and the greater national intelligence community.

The Director of the National Security Agency (DIRNSA), LT General Michael Hayden, has made it his personal goal to guide the nation's largest intelligence organization through a transitional period from the era of legacy-based systems and ideas to tomorrow's era of information technology-based intelligence exploitation. With the release of the National Security Agency's SIGINT Business Plan in March of this year, the foundations for a national cryptologic strategy for the 21st century were laid out. First and foremost among the many goals of this strategy is a shift in psychology, products and services to those of an information technology environment. This thesis is a critical, early step in the direction of fulfilling NSA's goals of expeditious transition.

B. WHY LINUX?

In the last few of years and, more importantly, over the last several months there has been explosion of interest in Linux as an advantageous alternative to Windows-based client/server operating systems. In fact, there is speculation that in the years to come, Linux will overtake the Microsoft share of operating systems in the global market. In spite of the February 2000 release of Windows 2000, a recent IDC study indicates that Linux is growing and will continue to grow at a much faster rate than other operating systems. (Loshin, 2000)



Source: IDC, August 1999 (www.idc.com)

Figure 1. Linux Growth From Ref [Loshin, 2000]

If considering the influence of the sheer volume of sales potential an operating system could have in the world market, one could turn to China's recent commissioning of a Linux-based operating system as an example. Compaq China has recently begun the development of a simplified Chinese Edition of the Linux operating system at the request of the Chinese government. Compaq China will team up with Beijing Founder Electronics and the Institute of Software Academia Sinica's Sino-Software System Company to develop what is to be known as "Red Flag Linux." The impetus for the development of the system will be to support the People's Republic of China government and Chinese enterprises in their initiatives to go online.

The decision to commission such a versatile operating system and provide True Type simplified Chinese display and printing capabilities has some at Microsoft up in arms. There is concern within Chinese government and industrial circles that heavy reliance on Microsoft 2000 (and other Microsoft operating systems) could lead to security leaks and make government computers more vulnerable to viruses. Additionally, Sun Yufang, an official overseeing the Red Flag software project at the Chinese Academy of Sciences reported that government offices expressed strong interest in scrapping Windows, citing concerns Microsoft had the ability to access their computers over the Internet. Of course, Microsoft representatives were quick to issue a statement of rebuttal insisting that the Chinese assessments were unsubstantiated media reports. Nevertheless, it appears that China is, at the very minimum, considering a viable alternative to the use of Microsoft-based client/server operating systems. (Staff, 1999)

In addition to China's burgeoning interest in Linux-based operating systems, several other countries such as Germany, Israel, Taiwan, Japan and even Australia have begun to pursue research in the acquisition of Linux systems tailored to meet their specific needs. (Staff, 2000) (Staff Reporter, 1999)

From a Cryptologic perspective, adversarial interest alone might be impetus enough to conduct research into the intricacies associated with the myriad security issues linked to web-based Linux systems. However, more than looking from a strictly exploitation point of view, one must consider the security issues associated with a defensive posture. As the global community, and more specifically, the United States continues to rely on the Internet and networking technologies, there is an ever-increasing necessity to stay one step ahead of the competition in terms of network defensive measures.

Recent network global attacks perpetrated by small groups or even individual hackers have raised the concern of world governments from guarded interest to all out paranoia. While investigations are still underway, it appears that the perpetrators of the Love Bug virus that swept around the world with alarming speed and devastation were a couple of Filipino college students. The virus, which spread through twenty countries and caused more than ten billion dollars in damage, is the most devastating computer bug in history. (Maney, 2000) It is, at least, the second such action to affect global network activities in the past year.

A few months ago, so called "friendly" forces, those whose self-proclaimed purpose it is to bring security vulnerabilities to the public's attention, struck against some of the largest web sites in the world like, Ebay, Yahoo, CNN, Amazon, E*Trade and DATEK. Electronic commerce slowed to a proverbial halt as the distributed denial of service attacks took their toll on America's net-based cyber trading and information exchange sites. (Levy/Stone, 2000) Attacks like these should be heeded as alarms that represent the potential that exists for adversarial governments or terrorist organizations to damage network systems. Their actions have caused the governments of the world to take a much closer look at how they allocate their resources with respect to network defense. The continued exponential growth and reliance on network-based operations prove that network security is a problem that will never fade away. Network defense will only continue to be the Achilles' heel for American industry, military and government.

It is for this reason that this research is focused on the Linux operating system. The primary goal of this research was to provide background and investigate the security issues associated with the Linux operating system. For this task, I chose the latest release of a popular Linux software package, Red Hat Linux 6.1. In this thesis I will describe the installation and creation of a Linux-based network, provide some insights into Linux system administration and report the findings of research which will test known security vulnerabilities and kernel patches recommended for those vulnerabilities in the inherent Linux firewall code, Ipchains.

C. INTRODUCTION

The Linux operating system was actually the brainchild of Linus Torvalds, an undergraduate student from the University of Helsinki in Finland. (Sobell, 1997) It's an offshoot of the original UNIX operating system that was developed in the mid-1970's. Over the years, various organizations had tailored the UNIX code to meet the specific needs of their association. This was a very tedious process, not to mention the costs of commercial UNIX, which were phenomenally high for the time. Torvalds developed Linux as an alternative to the commercial code, something that would be accessible to the average user at a very inexpensive price, free! While it was originally developed as a hobby, something that programmers could tinker with, it quickly evolved into a powerful kernel that entire teams of software developers modified for the sheer joy of having a hand in its development. By 1992, it was officially released as a complete, fully operational UNIX-like operating system. (Danesh, 1999)

When using the term "Linux", caution must be used so as not to cause confusion. In one sense, Linux refers to the software "kernel", the heart of any version of Linux, but in another sense, it is often used to refer to the current distribution - the collection of applications than run on the kernel. For deconfliction purposes, in this thesis, Linux will be used to refer to the operating system in general. "Kernel" and "distribution" will be used for clarification purposes. (Ziegler, 2000)

In addition to being an inexpensive alternative to Windows-based operating systems, the characteristics that make Linux such an attractive option lie in its attractive features such as multitasking and multi-user capability. Multitasking allows a computer with a single processor to appear to be performing multiple tasks simultaneously. While Windows systems like Windows 2000 and Windows NT are also multitasking, neither can compete with the robust multitasking capabilities Linux has to offer. Multi-user, as the word implies, allows multiple simultaneous users to log on to a network, thereby leveraging the multitasking capabilities of the operating system. This is a feature that has been inherent only to Unix operating systems until very recently with the release of Windows NT Terminal Server. (Shah, 2000)

Still, the average personal computer user and system administrator point to the ease of use of the graphical user interface (GUI) and the myriad of application software Windows systems have to offer. It turns out that Linux offers similar capabilities in their X Windows, built in scripting languages, word processing, databasing, DOS and Windows compatibility software all at a very reasonable price. The primary concern now is to investigate viable Linux-based security software for use by the average operator.

The principal concern when using any operating system in a network environment is its inherent security vulnerabilities. Linux is no more vulnerable than its Windows-based counterparts and, in fact, may offer added security measures not available in other systems. In addition to flat out, better built-in security prevention features, Linux also offers the advantage of "open source code", which provides the user with the option to find vulnerabilities and patch them at the root level. Windows-based systems require a complete new software release from the manufacturer as Microsoft's code is not available to the root user. (Danesh, 1999)

When considering security issues from a networking perspective the primary point of defense from possible adversarial users or organizations is the network firewall. As previously cited, the Linux firewall, ipchains, will be the focus of this thesis, but bear in mind that there are many commercial firewall options available to both the Linux and Windows NT system administrator.

II. FIREWALLS

As many people know, a firewall is found in every car. In a car, the firewall is a physical barrier (usually metal) that separates the engine from the passengers. It is meant to protect the passengers in the car from any dangers should the engine catch fire, while allowing the driver continued access to the engine's controls. A firewall in a computer provides a similar service in that it protects a private network from possible "dangers" in the public domain of the Internet. Dangers can come in the form of ill-intentioned individuals or groups who would like to penetrate an internal network. Usually, one computer in an internal network will act as the "firewall," protecting the internal network from outside penetration. However, there are limitations with firewalls, primarily that the protected network can't reach the external Internet directly, but by the same token, external users cannot reach the protected network without first going through the firewall.

A firewall is an attempt to provide security. It aids in the implementation of a larger security policy that defines the permitted access and services. It implements security policy through network configuration, the use of host systems and routers, and other security measures such as advanced authentication in place of static passwords. A firewall system's main purpose is to control access to or from a protected network (i.e., a site). It performs network access security by forcing connections to pass through the firewall, where they can be examined and evaluated.

A firewall system can be a router or a host (i.e. a stand-alone personal computer), set up specifically to shield a site or local area network from protocols and services that can be abused by hosts outside the local area network. A firewall system is usually located at a higher-level gateway, such as a site's connection to a wide area network (i.e., the Internet), however firewall systems can be at lower-level gateways and provide protection for some smaller collection of hosts or local area networks.

Firewalls can provide a measured level of confidence in network security, but are not the final answer in protecting a network. Firewalls cannot protect against attacks that emanate from within the network. Viruses are capable of penetrating firewalls. There are too many different methods of encoding binary files for transfer over networks and too many different viruses to protect from them all. Network users must be security conscious. Firewalls cannot prevent network users from allowing mail or copied files to be injected into and executed within the network causing data-driven attacks. Firewalls do an excellent job of keeping the outside world out of corporate networks but many research studies have shown that the threat from internal attackers is far more significant. (Cheswick, 1994)

A. OVERVIEW

A firewall puts up a barrier that controls the flow of traffic between networks. The safest firewall would block all traffic, but that defeats the purpose of making the connection, so the system administrator needs to strictly control selected traffic in a secure way. Firewalls are often described in terms of perimeter defense systems, with a so-called "choke point" through which all internal and external traffic is controlled. The usual metaphor is the medieval castle and its perimeter defense systems. The moats and walls provide the perimeter defense, while the gatehouses and drawbridges provide "choke points" through which everyone must travel to enter or leave the castle. Access can monitored and blocked at these choke points.

The real threat is often the stealthy spy who slips over the castle wall in the dark of night and scales every barrier undetected to reach his target of attack. The primary question for any system administrator is how far do you let people into the internal network, and what do you allow them to do once inside? In keeping with our castle analogy, local townspeople and traders were usually allowed to enter the market yard of the castle with relative ease so they could deliver or pick up goods. At night, the gates were closed, and goods were brought into the castle--usually after close inspection. Following this correlation, the market yard could be compared to the public Web and FTP servers that you connect to the Internet for general availability. While just about anybody could enter the market yard, only trusted people and people with special credentials were allowed into the inner perimeters of the castle. Like the multiple perimeter defenses of the castles, multiple firewall devices can be installed to keep devious hackers out of your networks. "Trip wires" can be set up by installing "relatively weak" devices on the outer edge of your defense that sound alarms when attacked. Once in place, a firewall, just as a castle, requires constant vigilance. If someone can climb your fence at night when no one is looking, what good is the fence? Security policies and procedures must be put into place. In defending a castle, the archers and boiling oil men need a defensive strategy, and they need regular drills to ensure that the strategy works. If your internal systems are hit by flaming arrows, you'll need a disaster recovery plan to put out the fire and get the system back online. Castle towers protect the soldiers who defend the castles. Without defenders, the castle is vulnerable to attackers that scale walls or knock down doors. Likewise, your firewall is not a stand-alone device. You need to manage and monitor it on a regular basis and to take action in the event of an attack. It is also only one part of your defense. If the attackers do get inside, you need to keep them from looting your systems by implementing security measures at each domain and server. This brings up another point. While firewalls are keeping Internet intruders out, your internal users might be looting your systems. You may need to separate departments, workgroups, divisions, or business partners using the same firewall technology, and you may need to implement encryption throughout your organization. Firewalls also do not protect against leaks, such as users connecting to the outside with a desktop modem. In addition, if some new threat comes along, your firewall might not be able to protect against it. Viruses and misuse of security devices are also a threat. (Cheswick, 1994)

B. DMZ

One answer to network security may be the use of a DMZ. "DMZ" is an abbreviation for "demilitarized zone". In the context of firewalls, this refers to a part of the network that is neither part of the internal network, nor directly part of the Internet. A DMZ can be created by putting access control lists on your access router. This minimizes the exposure of hosts on your external LAN by allowing only recognized and managed services on those hosts to be accessible by hosts on the Internet. These services are not required for the operation of a web server, so blocking Transmission Control Protocol (TCP) connections on that host will reduce the exposure to a denial-of-service attack. In fact, if you block everything but Hypertext Transfer Protocol (HTTP) traffic to that host, an attacker will only have one service to attack.

A common approach for an attacker is to break into a host that's vulnerable to attack, and exploit trusts relationships between the vulnerable host and more interesting targets. If you are running a number of services that have different levels of security, you might want to consider breaking your DMZ into several "security zones". This can be done by having a number of different networks within the DMZ. For example, the access router could feed two ethernets.

On one of the ethernets, you might have hosts whose purpose is to service your organization's need for Internet connectivity. These will likely relay mail, news, and host DNS. On the other Ethernet could be your web server(s) and other hosts that provide services for the benefit of Internet users.

In many organizations, services for Internet users tend to be less carefully guarded and are more likely to be doing insecure things. This might be reasonable for your web server, but brings with it a certain set of risks that need to be managed. It is likely that these services are too risky for an organization to run them on a bastion host, where a slip-up can result in the complete failure of the security mechanisms.

By putting hosts with similar levels of risk on networks together in the DMZ, you can help minimize the effect of a break in at your site. If someone breaks into your web server by exploiting some bug in your web server, they won't be able to use it as a launching point to break into your private network if the web servers are on a separate LAN from the bastion hosts and you don't have any trust relationships between the web server and bastion host.

Keep in mind that we're running Ethernet here. If someone breaks into your web server, and your bastion host is on the same Ethernet, an attacker can install a sniffer on your web server, and watch the traffic to and from your bastion host. This might reveal things that can be used to break into the bastion host and gain access to the internal network.

Splitting services up not only by host, but also by network, and limiting the level of trust between hosts on those networks, you can greatly reduce the likelihood of a break in on one host being used to break into the other. In other words, breaking into the web server in this case won't make it any easier to break into the bastion host. In this way, routers become the first lines of defense in a comprehensive security strategy.

You can also increase the scalability of your architecture by placing hosts on different networks. The fewer machines that share the available bandwidth, the more bandwidth that each will get. (Cheswick, 1994)

III. TYPES OF FIREWALLS

A. OSI MODEL

In keeping with the interoperability goals as outlined by the Unified Cryptologic Architecture and mandated by NSA in May 2000, it is imperative that network security be given the highest priority. Since routers are considered to be a first line of defense for most networks, prioritization requires that a thorough understanding of how routers can be used as firewalls must be undertaken. Hence, a discussion of the various types of router-based firewalls ensues.

First, an introduction of the Open Systems Interconnection (OSI) network model is appropriate. The network model is comprised of seven layers, referred to as OSI. OSI provides for the transfer of information between end systems across some sort of communications network. It relieves higher layers of the need to know about the underlying data transmission and switching technologies used to connect systems. The lower three layers are concerned with attaching to and communicating with the network, while the others are concerned with the specific functions of data exchange.

OSI Model	
Layer 7:	Application Layer
Layer 6:	Presentation Layer
Layer 5:	Session Layer
Layer 4:	Transport Layer
Layer 3:	Network Layer
Layer 2:	Data Link Layer
Layer 1:	Physical Layer

Table 1. OSI Model [Stallings, 1997]

the port is listening. A RST is indicative of a non-listener. If a SYN|ACK is received, a RST is immediately sent to tear down the connection (actually our OS kernel does this for us). The primary advantage to this scanning technique is that fewer sites will log it. Unfortunately you need root privileges to build these custom SYN packets.

-sF -sX –sN

Stealth FIN, Xmas Tree, or Null scan modes: There are times when even SYN scanning isn't clandestine enough. Some firewalls and packet filters watch for SYNs to restricted ports, and programs like Synlogger and Courtney are available to detect these scans. These advanced scans, on the other hand, may be able to pass through unmolested. The idea is that closed ports are required to reply to your probe packet with an RST, while open ports must ignore the packets in question (see RFC 793 pp 64). The FIN scan uses a bare (surprise) FIN packet as the probe, while the Xmas tree scan turns on the FIN, URG, and PUSH flags. The Null scan turns off all flags. Unfortunately Microsoft (like usual) decided to completely ignore the standard and do things their own way. Thus this scan type will not work against systems running Windows95/NT. On the positive side, this is a good way to distinguish between the two platforms. If the scan finds open ports, you know the machine is not a Windows box. If a -sF,-sX, or -sN scan shows all ports closed, yet a SYN (-sS) scan shows ports being opened, you are probably looking at a Windows box. This is less useful now that nmap has proper OS detection built in. There are also a few other systems that are broken in the same way Windows is. They include Cisco, BSDI, HP/UX, MVS, and IRIX. All of the above send resets from the open ports when they should just drop the packet.

-sP Ping scanning: Sometimes you only want to know which hosts on a network are up. Nmap can do this by sending ICMP echo request packets to every IP address on the networks you specify. Hosts that respond are up. Unfortunately, some sites such as microsoft.com block echo request packets. Thus nmap can also send a TCP ack packet to (by default) port 80. If we get an RST back, that machine is up. A third technique involves sending a SYN packet and waiting for a RST or a SYN/ACK. For non-root users, a connect() method is used. By default (for root users), nmap uses both the ICMP and ACK techniques in parallel. You can change the **-P** option described later. Note that pinging is done by default anyway, and only hosts that respond are scanned. Only use this option if you wish to ping sweep **without** doing any actual port scans.

-sU UDP scans: This method is used to determine which UDP (User Datagram Protocol, RFC 768) ports are open on a host. The technique is to send 0 byte udp packets to each port on the target machine. If we receive an ICMP port unreachable message, then the port is closed. Otherwise we assume it is open. Some people think UDP scanning is pointless. I usually remind them of the recent Solaris rcpbind hole. Rpcbind can be found hiding on an undocumented UDP port somewhere above 32770. So it doesn't matter that 111 is blocked by the firewall. But can you find which of the more than 30,000 high ports it is listening on? With a UDP scanner you can! There is also the cDc Back Orifice backdoor program, which hides on a configurable UDP port on Windows machines. Not to mention the many commonly vulnerable services that utilize UDP such as snmp, tftp, NFS, etc. Unfortunately UDP scanning is sometimes painfully slow since most hosts implement a suggestion in RFC 1812 (section 4.3.2.8) of limiting the ICMP error message rate. For example, the Linux kernel (innet/ipv4/icmp.h) limits destination unreachable message generation to 80 per 4 seconds, with a 1/4 second penalty if that is exceeded. Solaris has much more strict limits (about 2 messages per second) and thus takes even longer to scan. *nmap* detects this rate limiting and slows down accordingly, rather than flood the network with useless packets that will be ignored by the target machine. As is typical, Microsoft ignored the suggestion of the RFC and does not seem to do any rate limiting at all on Win95 and NT machines. Thus we can scan all 65K ports of a Windows machine **very** quickly.

-sA ACK scan: This advanced method is usually used to map out firewall rule sets. In particular, it can help determine whether a firewall is stateful or just a simple packet filter that blocks incoming SYN packets. This scan type sends an ACK packet (with random looking acknowledgement/sequence numbers) to the ports specified. If a RST comes back, the ports is classified as "unfiltered". If nothing comes back (or if an ICMP unreachable is returned), the port is classified as "filtered". Note that *nmap* usually doesn't print "unfiltered" ports, so getting **no** ports shown in the output is usually a sign that all the probes got through (and returned RSTs). This scan will obviously ever show ports in the ”open” state.

-sW Window scan: This advanced scan is very similar to the ACK scan, except that it can sometimes detect open ports as well as filtered/nonfiltered due to anomaly in the TCP window size reporting by some operating systems. Systems vulnerable to this include at least some versions of AIX, Amiga, BeOS, BSDI, Cray, Tru64 UNIX, DG/UX, OpenVMS, Digital UNIX, FreeBSD, HP-UX, OS/2, IRIX, MacOS, NetBSD, OpenBSD, OpenStep, QNX, Rhapsody, SunOS 4.X, Ultrix, VAX, and VxWorks. See the nmap-hackers mailing list archive for a full list.

-sR RPC scan. This method works in combination with the various port scan methods of Nmap. It takes all the TCP/UDP ports found open and then floods them with SunRPC program NULL commands in an attempt to determine whether they are RPC ports, and if so, what program and version number they serve up. Thus you can effectively obtain the same info as firewall (or protected by TCP wrappers). Decoys do not currently work with RPC scan, at some point I may add decoy support for UDP RPC scans.

-b <ftp relay host> FTP bounce attack: An interesting "feature" of the ftp protocol (RFC 959) is support for "proxy" ftp connections. In other words, I should be able to connect from evil.com to the FTP server of target.com and request that the server end a file ANYWHERE on the internet! Now this may have worked well in 1985 when the RFC was written. But in today's Internet, we can't have people hijacking ftp servers and requesting that data be spit out to arbitrary points on the internet. As *Hobbit* wrote back in 1995, this protocol flaw "can be used to post virtually untraceable mail and news, hammer on servers at various sites, fill up disks, try to hop firewalls, and generally be annoying and hard to track down at the same time." What we will exploit this for is to scan TCP ports from a "proxy" ftp server. Thus you could connect to an ftp server behind a firewall, and then scan ports that are more likely to be blocked (139 is a good one). If the ftp server allows reading from and writing to some directory (such as /incoming), you can send arbitrary data to ports that you do find open (nmap doesn't do this for you though). The argument passed to the 'b' option is the host you want to use as a proxy, in standard URL notation. The format is: *username:password@server:port.* Everything but *server* is optional.

GENERAL OPTIONS

None of these options are required but some can be quite useful.

-P0 Do not try and ping hosts at all before scanning them. This allows the scanning of networks that don't allow ICMP echo requests (or responses) through their firewall. microsoft.com is an example of such a network, and thus you should always use **-P0** or **-PT80** when portscanning microsoft.com.

-PT Use TCP "ping" to determine what hosts are up. Instead of sending ICMP echo request packets and waiting for a response, we spew out TCP ACK packets throughout the target network (or to a single machine) and then wait for responses to trickle back. Hosts that are up should respond with a RST. This option preserves the efficiency of only scanning hosts that are up while still allowing you to scan networks/hosts that block ping packets. For non root users, we use connect(). To set the destination port of the probe packets use -PT<port number>. The default port is 80, since this port is often not filtered out.

-PS This option uses SYN (connection request) packets instead of ACK packets for root users. Hosts that are up should respond with a RST (or, rarely, a SYN|ACK).

-PI This option uses a true ping (ICMP echo request) packet. It finds hosts that are up and also looks for subnet-directed broadcast addresses on your network. These are IP addresses, which are externally reachable and translate to a broadcast of incoming IP packets to a subnet of computers. These should be eliminated if found as they allow for numerous denial of service attacks (Smurf is the most common).

-PB This is the default ping type. It uses both the ACK (**-PT**) and ICMP (**-IP**) sweeps in parallel. This way you can get firewalls that filter either one (but not both).

-O This option activates remote host identification via TCP/IP fingerprinting. In other words, it uses a bunch of techniques to detect subtleties in the underlying operating system network stack of the computers you are scanning. It uses this information to create a 'fingerprint' which it compares with its database of known OS fingerprints (the nmap-os-fingerprints file) to decide what type of system you are scanning. If you can't send the IP address, the next best thing is to run nmap with the **-d** option and send me the three fingerprints that should result along with the OS name and version number. By doing this you contribute to the pool of operating systems known to nmap and thus it will be more accurate for everyone.

-I This turns on TCP reverse ident scanning. As noted by Dave Goldsmith in a 1996 Bugtraq post, the ident protocol (rfc 1413) allows for the disclosure of the username that owns any process connected via TCP, even if that process didn't initiate the connection. So, you can, for example, connect to the http port and then use identd to find out whether the server is running as root. This can only be done with a full TCP connection to the target port (i.e. the -sT scanning option). When **-I** is used, the remote host's identd is queried for each open port found. Obviously this won't work if the host is not running identd.

-f This option causes the requested SYN, FIN, XMAS, or NULL scan to use tiny fragmented IP packets. The idea is to split up the TCP header over several packets to make it harder for packet filters, intrusion detection systems, and other annoyances to detect what you are doing. Be careful with this! Some programs have trouble handling these tiny packets. My favorite sniffer segmentation faulted immediately upon receiving the first 36-byte fragment. After that comes a 24 byte one! While this method won't get by packet filters and firewalls that queue all IP fragments (like the CONFIG_IP_ALWAYS_DEFRAG option in the Linux kernel), some networks can't afford the performance hit this causes and thus leave it disabled. Note that I do not yet have this option working on all systems. It works fine for my Linux, FreeBSD, and OpenBSD boxes and some people have reported success with other *NIX variants.

-v Verbose mode. This is a highly recommended option and it gives out more information about what is going on. You can use it twice for greater effect. Use **-d** a couple of times if you really want to get crazy with scrolling the screen!

-h This handy option display a quick reference screen of nmap usage options. As you may have noticed, this man page is not exactly a 'quick reference'.

-oN <logfilename> This logs the results of your scans in a normal human readable form into the file you specify as an argument.

-oM <logfilename> This logs the results of your scans in a **machine parseable** form into the file you specify as an argument. You can give the argument '-' (without quotes) to shoot output into stdout (for shell pipelines, etc). In this case normal output will be suppressed. Watch out for error messages if you use this (they will still go to stderr). Also note that '-v' will cause some extra information to be printed.

--resume <logfilename> A network scan that is cancelled due to control-C, network outage, etc. can be resumed using this option. The logfilename must be either a normal (-oN) or machine parsable (-oM) log from the aborted scan. No other options can be given (they will be the same as the aborted scan). Nmap will start on the machine after the last one successfully scanned in the log file.

-iL <inputfilename> Reads target specifications from the file specified RATHER than from the command line. The file should contain a list of host or network expressions separated by spaces, tabs, or newlines. Use a hyphen (-) as *inputfilename* if you want nmap to read host expressions from stdin (like at the end of a pipe). See the section *target specification* for more information on the expressions you fill the file with.

-iR This option tells Nmap to generate its own hosts to scan by simply picking random numbers. It will never end. This can be useful for statistical sampling of the Internet to estimate various things. If you are ever really bored, try *nmap -sS -iR -p 80* to find some web servers to look at.

-p <port ranges> This option specifies what ports you want to specify. For example '-p 23' will only try port 23 of the target host(s). '-p 20-30,139,60000-' scans ports between 20 and 30, port 139, and all ports greater than 60000. The default is to scan all ports between 1 and 1024 as well as any ports listed in the services file which comes with nmap.

-F Fast scan mode.

Specifies that you only wish to scan for ports listed in the services file which comes with nmap. This is obviously much faster than scanning all 65535 ports on a host.

-D <decoy1 [,decoy2][,ME],...> Causes a decoy scan to be performed which makes it appear to the remote host that the host(s) you specify as decoys are scanning the target network too. Thus their IDS might report 5-10 port scans from unique IP addresses, but they won't know which IP was scanning them and which were innocent decoys. While this can be defeated through router path tracing, response-dropping, and other "active" mechanisms, it is generally an extremely effective technique for hiding your IP address. Separate each decoy host with commas, and you can optionally use 'ME' as one of the decoys to represent the position you want your IP address to be used. If your put 'ME' in the 6th position or later, some common port scan detectors (such as Solar Designer's excellent scanlogd) are unlikely to show your IP address at all. If you don't use 'ME', nmap will put you in a random position. Note that the hosts you use as decoys should be up or you might accidentally SYN flood your targets. Also it will be pretty easy to determine which host is scanning if only one is actually up on the network. You might want to use IP addresses instead of names (so the decoy networks don't see you in their nameserver logs). Also note that some (stupid) "port scan detectors" will firewall/deny routing to hosts that attempt port scans. Thus you might inadvertently cause the machine you scan to lose connectivity with the decoy machines you are using. This could cause the target machines major problems if the decoy is, say, its internet gateway or even "localhost". Thus you might want to be careful of this option. The real moral of the story is that detectors of spoofable port scans should not take action against the machine that seems like it is port scanning them. It could just be a decoy! Decoys are used both in the initial ping scan (using ICMP, SYN, ACK, or whatever) and during the actual port scanning phase. Decoys are also used during remote OS detection (**-O**). It is worth noting that using too many decoys may slow your scan and potentially even make it less accurate. Also, some ISPs will filter out your spoofed packets, although many (currently most) do not restrict spoofed IP packets at all.

-S <IP_Address> In some circumstances, *nmap* may not be able to determine your source address (*nmap* will tell you if this is the case). In this situation, use -S with your IP address (of the interface you wish to send packets through). Another possible use of this flag is to spoof the scan to make the targets think that **someone else** is scanning them. Imagine a company being repeatedly port scanned by a competitor! This is not a supported usage (or the main purpose) of this flag. I just think it raises an interesting possibility that people should be aware of before they go accusing others of port scanning them. **-e** would generally be required for this sort of usage.

-e <interface> Tells nmap what interface to send and receive packets on. Nmap should be able to detect this but it will tell you if it cannot.

-g <portnumber> Sets the source port number used in scans. Many naive firewall and packet filter installations make an exception in their rule set to allow DNS (53) or FTP-DATA (20) packets to come through and establish a connection. Obviously this completely subverts the security advantages of the firewall since intruders can just masquerade as FTP or DNS by modifying their source port. Obviously for a UDP scan you should try 53 first and TCP scans should try 20 before 53. Note that this is only a request nmap will honor it only if and when it is able to. For example, you can't do TCP ISN sampling all from one host: port to one host: port, so nmap changes the source port even if you used -g. Be aware that there is a small performance penalty on some scans for using this option, because I sometimes store useful information in the source port number.

-r Tells Nmap **NOT** to randomize the order in which ports are scanned.

--randomize_hosts Tells Nmap to shuffle each group of up to 2048 hosts before it scans them. This can make the scans less obvious to various network monitoring systems, especially when you combine it with slow timing options (see below).

-M <max sockets> Sets the maximum number of sockets that will be used in parallel for a TCP connect() scan (the default). This is useful to slow down the scan a little bit and avoid crashing remote machines. Another approach is to use -sS, which is generally easier for machines to handle.

TIMING OPTIONS

Generally Nmap does a good job at adjusting for Network characteristics at runtime and scanning as fast as possible while minimizing that chances of hosts/ports going undetected. However, there are same cases where Nmap's default timing policy may not meet your objectives. The following options provide a fine level of control over the scan timing:

-T <Paranoid|Sneaky|Polite|Normal|Aggressive|Insane> These are canned timing policies for conveniently expressing your priorities to Nmap. **Paranoid** mode scans **very** slowly in the hopes of avoiding detection by IDS systems. It serializes all scans (no parallel scanning) and generally waits at least 5 minutes between sending packets. **Sneaky** is similar, except it only waits 15 seconds between sending packets. **Polite** is meant to ease load on the network and reduce the chances of crashing machines. It serializes the probes and waits **at least** 0.4 seconds between them.

Normal is the default Nmap behavior, which tries to run as quickly as possible without overloading the network or missing hosts/ports. **Aggressive** mode adds a 5 minute timeout per host and it never waits more than 1.25 seconds for probe responses. **Insane** is only suitable for very fast networks or where you don't mind losing some information. It times out hosts in 75 seconds and only waits 0.3 seconds for individual probes. It does allow for very quick network sweeps though :). You can also reference these by number (0-5). For example, '-T 0' gives you Paranoid mode and '-T 5' is Insane mode. These canned timing modes should NOT be used in combination with the lower level controls given below.

--host_timeout <milliseconds> Specifies the amount of time Nmap is allowed to spend scanning a single host before giving up on that IP. The default timing mode has no host timeout.

--max_rtt_timeout <milliseconds> Specifies the maximum amount of time Nmap is allowed to wait for a probe response before retransmitting or timing out that particular probe. The default mode sets this to about 9000.

--min_rtt_timeout <milliseconds> When the target hosts start to establish a pattern of responding very quickly, Nmap will shrink the amount of time given per probe. This speeds up the scan, but can lead to missed packets when a response takes longer than usual. With this parameter you can guarantee that Nmap will wait at least the given amount of time before giving up on a probe.

--initial_rtt_timeout <milliseconds> Specifies the initial probe timeout. This is generally only useful when scanning firewalled hosts with -P0. Normally Nmap can obtain good RTT estimates from the ping and the first few probes. The default mode uses 6000.

--max_parallelism <number> Specifies the maximum number of scans Nmap is allowed to perform in parallel. Setting this to one means Nmap will never try to scan more than one port at a time. It also effects other parallel scans such as ping sweep, RPC scan, etc.

--scan_delay <milliseconds> Specifies the **minimum** amount of time Nmap must wait between probes. This is mostly useful to reduce network load or to slow the scan way down to sneak under IDS thresholds. (Granquist, 1999)

APPENDIX C: GLOSSARY

Abuse of Privilege:

When a user performs an action that they should not have, according to organizational policy or law.

Access Control Lists:

Rules for packet filters (typically routers) that define which packets to pass and which to block.

Access Router:

A router that connects your network to the external Internet. Typically, this is your first line of defense against attackers from the outside Internet. By enabling access control lists on this router, you'll be able to provide a level of protection for all of the hosts "behind" that router, effectively making that network a DMZ instead of an unprotected external LAN.

Application-Level Firewall:

A firewall system in which service is provided by processes that maintain complete TCP connection state and sequencing. Application level firewalls often re-address traffic so that outgoing traffic appears to have originated from the firewall, rather than the internal host.

Authentication:

The process of determining the identity of a user that is attempting to access a system.

Authentication Token:

A portable device used for authenticating a user. Authentication tokens operate by challenge/response, time-based code sequences, or other techniques. This may include paper-based lists of one-time passwords.

Authorization:

The process of determining what types of activities are permitted. Usually, authorization is in the context of authentication: once you have authenticated a user, they may be authorized different types of access or activity.

Bastion Host:

A system that has been hardened to resist attack, and which is installed on a network in such a way that it is expected to potentially come under attack. Bastion hosts are often components of firewalls, or may be "outside" Web servers or public access systems. Generally, a bastion host is running some form of general purpose operating system (e.g., Unix, VMS, NT, etc.) rather than a ROM-based or firmware operating system.

Challenge/Response:

An authentication technique whereby a server sends an unpredictable challenge to the user, who computes a response using some form of authentication token.

Chroot:

A technique under Unix whereby a process is permanently restricted to an isolated subset of the file system.

Cryptographic Checksum:

A one-way function applied to a file to produce a unique "fingerprint" of the file for later reference. Checksum systems are a primary means of detecting file system tampering on Unix.

Data Driven Attack:

A form of attack in which the attack is encoded in innocuous-seeming data which is executed by a user or other software to implement an attack. In the case of firewalls, a data driven attack is a concern since it may get through the firewall in data form and launch an attack against a system behind the firewall.

Defense in Depth:

The security approach whereby each system on the network is secured to the greatest possible degree. May be used in conjunction with firewalls.

DNS spoofing:

Assuming the DNS name of another system by either corrupting the name service cache of a victim system, or by compromising a domain name server for a valid domain.

Dual Homed Gateway:

A dual homed gateway is a system that has two or more network interfaces, each of which is connected to a different network. In firewall configurations, a dual homed gateway usually acts to block or filter some or all of the traffic trying to pass between the networks.

Encrypting Router:

see Tunneling Router and Virtual Network Perimeter.

Firewall:

A system or combination of systems that enforces a boundary between two or more networks.

Host-based Security:

The technique of securing an individual system from attack. Host based security is operating system and version dependent.

Insider Attack:

An attack originating from inside a protected network.

Intrusion Detection:

Detection of break-ins or break-in attempts either manually or via software expert systems that operate on logs or other information available on the network.

IP Spoofing:

An attack whereby a system attempts to illicitly impersonate another system by using its IP network address.

IP Splicing / Hijacking:

An attack whereby an active, established, session is intercepted and co-opted by the attacker. IP Splicing attacks may occur after an authentication has been made, permitting the attacker to assume the role of an already authorized user. Primary protections against IP Splicing rely on encryption at the session or network layer.

Least Privilege:

Designing operational aspects of a system to operate with a minimum amount of system privilege. This reduces the authorization level at which various actions are performed and decreases the chance that a process or user with high privileges may be caused to perform unauthorized activity resulting in a security breach.

Logging:

The process of storing information about events that occurred on the firewall or network.

Log Retention:

How long audit logs are retained and maintained.

Log Processing:

How audit logs are processed, searched for key events, or summarized.

Network-Level Firewall:

A firewall in which traffic is examined at the network protocol packet level.

Perimeter-based Security:

The technique of securing a network by controlling access to all entry and exit points of the network.

Policy:

Organization-level rules governing acceptable use of computing resources, security practices, and operational procedures.

Proxy:

A software agent that acts on behalf of a user. Typical proxies accept a connection from a user, make a decision as to whether or not the user or client IP address is permitted to use the proxy, perhaps does additional authentication, and then completes a connection on behalf of the user to a remote destination.

Screened Host:

A host on a network behind a screening router. The degree to which a screened host may be accessed depends on the screening rules in the router.

Screened Subnet:

A subnet behind a screening router. The degree to which the subnet may be accessed depends on the screening rules in the router.

Screening Router:

A router configured to permit or deny traffic based on a set of permission rules installed by the administrator.

Session Stealing:

See IP Splicing.

Trojan Horse:

A software entity that appears to do something normal but which, in fact, contains a trapdoor or attack program.

Tunneling Router:

A router or system capable of routing traffic by encrypting it and encapsulating it for transmission across an untrusted network, for eventual de-encapsulation and decryption.

Social Engineering:

An attack based on deceiving users or administrators at the target site. Social engineering attacks are typically carried out by telephoning users or operators and pretending to be an authorized user, to attempt to gain illicit access to systems.

Virtual Network Perimeter:

A network that appears to be a single protected network behind firewalls, which actually encompasses encrypted virtual links over untrusted networks.

Virus:

A replicating code segment that attaches itself to a program or data file. Viruses might or might not contain attack programs or trapdoors. Unfortunately, many have taken to calling *any* malicious code a "virus". If you mean "Trojan horse" or "worm", say "Trojan horse" or "worm".

Worm:

A stand alone program that, when run, copies itself from one host to another, and then runs itself on each newly infected host. The widely reported "Internet Virus" of 1988 was not a virus at all, but actually a worm.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

1. Anonymous, Maximum Linux Security, pp. 487-509, Sams Publishing, Indianapolis, IN, 2000

2. Barkakati, Naba, Red Hat Linux Secrets, pp. 569, 621, 623, IDG Books Worldwide, Foster City, CA, 1999

3. Brenton, C., Mastering Network Security, Network Press, San Francisco, California, 1999

4. Breyer, Robert and Riley, Sean, Switched and Fast Ethernet: How It Works and How to Use It, pp. 174-211, Ziff-Davis Press, Emeryville, CA, 1995

5. Cheswick, William R. and Bellovin, Steven M., Firewalls and Internet Security and Internet Security: Repelling the Wiley Hacker, pp. 271-277, Addison~Wesley Publishing, Menlo Park, CA, 1994

6. Creed, Adam, LinuxOne Expands into Taiwan, Daily News, p. 1, January 6, 2000

7. Danesh, Arman, Red Hat Linux 6, pp. 179-187, 656-658,Sybex, San Francisco, CA, 1999

8. Edwards, Mark Joseph, Internet Security With Windows NT, pp. 25-33, Duke Communications Worldwide, Loveland, CO, 1998

9. Escamilla, Terry, Intrusion Detection , pp. 131-135, Wiley Computer Publishing, New York, NY, USA, 1998

10. Goncalves, Marcus, Protecting Your Web Site With Firewalls, pp. 81-84, Prentice Hall, Upper Saddle River, NJ, USA, 1997

11. Granquist, Lamont, Nmap Tutorial Guide, http//www.insecure.com/nmap/lamont-nmap-guide.txt, 5 April 99

12. Hallberg, Bruce, Networking A Beginner's Guide, pp. 100-112, Osborne/McGraw Hill, Berkeley, CA, 2000

13. Honegger, Barbara, NPS Named Center of Academic Excellence for INFOSEC, Campus News, Vol. 6 Issue 19, p. 1, May 11, 2000

14. Kyas, Othmar, Internet Security, pp. 18-22, International Thompson Publishing, Boston, MA, USA, 1997

15. Levy, Steven & Stone, Brad, Hunting the Hackers, Newsweek, pp. 38-44, February 21, 2000

16. Lowe, Doug, Networking for Dummies, pp. 213-214, 300, IDG Books Worldwide, Foster City, CA, 1999

17. Loshin, Pete, Securing Linux, Information Security, pp. 20-31, Feb 2000

18. Maney, Kevin & McMahon, Patrick, 'Love Bug' Virus Created in Ordinary Petri Dish, USA Today, p. 2, May 15, 2000

19. McClure, Stuart & Scambray, Joel, Kurtz, George, Hacking Exposed Network Security Secrets & Solutions, pp. 4-85, Osborne/McGraw Hill, Berkeley, CA, 1999

20. Schwoerer, Sean, Linux as a Firewall Solution?, Firewalls 101: Perimeter Protection With Firewalls, SANS Institute, Network Security Conference, San Jose, CA, pp. 4-1-4-55, May 9, 2000

21. Shah, Steve, Linux Administration A Beginner's Guide, pp. 476-483, Osborne/McGraw Hill, Berkeley, CA, 2000

22. Siyan, Karanjit & Ware, Chris, Internet Firewalls and Network Security, pp. 91-96, New Riders Publishing, Indianapolis, IN, 1995

23. Sobell, Mark G., A Practical Guide to Linux, pp. 3-17, Addison-Wesley, Berkeley, CA, 1997

24. Staff, Compaq to Develop Chinese Linux, IT Daily, [http://computercurrents.com/newstoday/99/09/01/news4.html]. Sept 1, 1999

25. Staff, Red Hat Expands into Japan, Daily News, [http://www.currents.net/news/99/09/09/news9.html]. Sept 9, 1999

26. Staff Reporter, Linux to Pay Off, Daily News, [http://www.currents.net/news/00/03/10/news9.html]. March 10, 2000

27. Stallings, William, Data and Computer Communications (Fifth Edition), pp. 211-216, Prentice Hall, Upper Saddle River, NJ, 1997

28. Stallings William, Internet Security Handbook, pp. 45-49, IDG Books Worldwide, Foster City, CA, 1995
29. Witherspoon, Coletta and Craig, Red Hat Linux 6: fast & easy, pp. 170-173, 142-151, Prima Tech, Rocklin, CA, 1999

30. Ziegler, Robert, Linux Firewalls, pp. 62-126, New Riders Publishing, Indianapolis, IN, 2000

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center...2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, Virginia 22060-6218

2. Dudley Knox Library...2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, California 93943-5101

3. Raymond Bernstein, Code/ECBe..1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California 93943-5121

4. Vicente Garcia, Code/ECGa...1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California 93943-5121

5. Chairman, Code EC..1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California 93943-5121

6. Bryan S. Lopez...2
   217 S. Tornillo St.
   Las Cruces, NM 88001

66 290NPG 2745
TH
6/02 22527-200 NLB

DUDLEY KNOX LIBRARY
3 2768 00403849 7